The Star Solution

Komi Agbo, Dalton Burke, Nick Mako, James Vance

November 23, 2019

Abstract

Sam's Hauling, Inc. provides waste disposal containers throughout the Denver metro area. In order to increase efficiency they are seeking a way to create delivery schedules for their drivers, subject to a zoo of constraints. Our solution averts standard Traveling Salesman Problem style approach for something tailored specifically for the problem. The result is a fast, nearly optimal solution which takes advantage of how central Landfills and Storage Yards are to the problem.

Keywords - Optimization, Constrained VRP, Assignment Problem

1 Introduction

Sam's Hauling, Inc. provides roll-off dumpsters (colloquially, *cans*) of various sizes to homeowners, contractors, realtors and property managers throughout the Metro Denver area. Each job site (delivery, pickup, or switch), has some constraints, they need a particular can size, they often need it at a certain time (AM, PM, or unrestricted), sometimes they need a certain sized truck to do it. They have a limited number of trucks with which to do these pickups and deliveries and in addition some customers set time windows for said pickups and deliveries. There are multiple depots where they store the cans, which start the day with a certain number of cans in stock.

At first blush, the problem seems to be some nightmarish amalgamation of complicated constraints and NP-Complete[citation] problems, however, its saving grace is that trucks can carry only one container at a time, and must return to a landfill or a storage yard between jobs frequently.

Currently, the pickup and delivery of these cans is scheduled by a single individual using only Microsoft Excel. This method for solving it is very time intensive and is less likely to be the most efficient solution. It is quite likely then, that the implementation of some heuristic or metaheuristic could increase the efficiency of this process by a significant margin.

2 Background

The Sam's Hauling Problem (SHP) initially seems very similar to the Traveling Salesman Problem[citation] which asks the question, "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?" If the TSP had multiple salesmen, (as in the Vehicle Routing Problem[citation], VRP), we'd need to figure out a route for each driver so that they visit every job, and return to the home base at the end of the day.

The parallels are obvious, however, the biggest problem with the TSP and the VRP is that they are both NP-Complete, which makes them computationally infeasible to find optimal solutions for. To solve these problems requires heuristic methods, which can still be computationally expensive, and will almost certainly not give an optimal solutions.

Our approach doesn't depend very much on known algorithms, outside of one, the Hungarian Algorithm, which is a polynomial time algorithm ($O(n^3)$) that generates an optimal solution to a matching problem[citation]. In our application, we wanted to pair up deliveries and pickups so that the total route cost (over all drivers) was minimized.

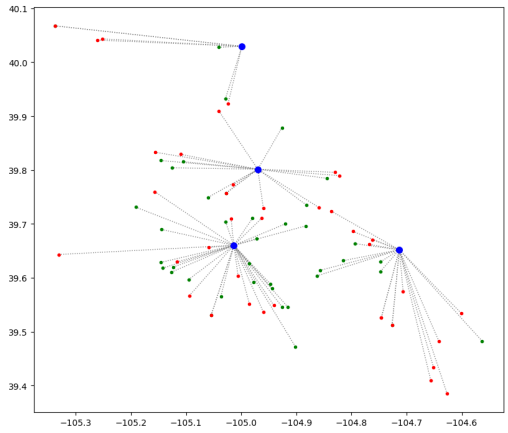
3 Methods

Instead of thinking of the problem as a Vehicle Routing Problem, where the order of deliveries directly effects the cost of the route, our team noticed that

how much the landfills and storage locations are visited nearly eliminates this ordering constraint.

3.1 Stars

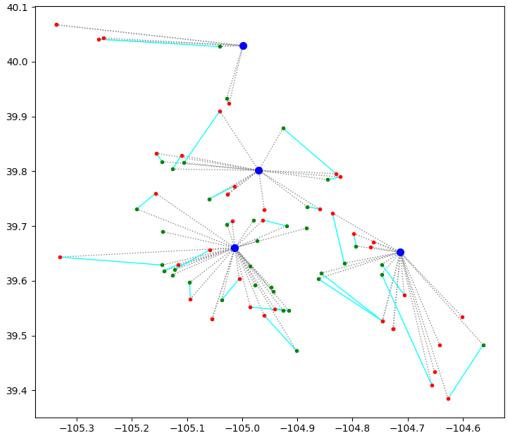
If customers only ever wanted container switches, the problem would be much simpler. Where should the truck come from that's going to deliver the new container? The nearest landfill. And where should the truck go after picking up the full container? The nearest landfill. Filling the route with this data, we end up with several *stars*, each landfill being a center, and the points surrounding it being the job sites. To service each star, you can take the jobs in any order, the resulting cost will be the same. This will give us a lot of leniency when trying to satisfy the constraints. The only task remaining is to sew them together into routes for the drivers.



Blue dots are landfills/storage yards, green dots are pickups, red dots are deliveries.

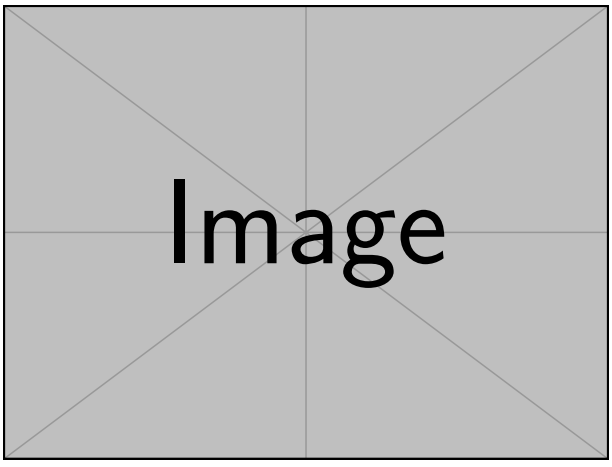
3.2 Triangles

Since the problem doesn't actually consist solely of switches it is necessary to expand the star solution to include pickup and drop-offs. This provides a route of landfill, drop-off, pick-up, landfill for a driver, creating a triangle. These triangle routes allow for further optimization as long as the optimal triangles are chosen.



Triangles

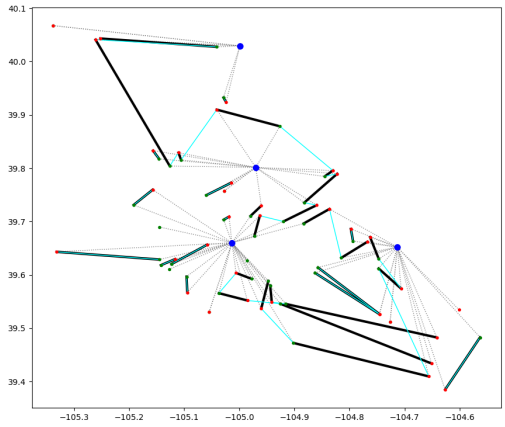
To make this decision, we can construct a bipartite graph, one set containing the drop-offs, and the other containing pick-ups, and the existence of an edge between a drop-off and pick-up means that they are compatible (their constraints can work together). The weight of the edge between a drop-off and a pick-up is calculated as the difference between doing each normally (star method) and doing them combined as a triangle. In this graph, a matching pairs up drop-offs and pick-ups to be completed as a triangle. A minimum matching in this graph gives us the best possible choice of triangles in order to make the route take the least amount of time.



A matching in a mathematical sense

We allow for pairs to cross between landfill zones, that is, the delivery has a closest landfill L_1, and the pickup has a different nearest landfill. We allow the

two to be matched up, the resulting matching always at least as well (almost always better) as a matching where the delivery and pickup must have the same nearest landfill. We also found the constraints to be just as easy to satisfy.



Deliveries and Pickups matched from real data. Cyan edges are a part of the matching where deliveries and pickups must have the same nearest landfill, and black edges are a part of the matching where they are not restricted.

The optimal matching can be calculated in polynomial time by the Hungarian Algorithm. In our implementation, we specifically used the Munkres package in Python.

3.3 Route Building

Now that we have matched up deliveries and pickups optimally, we need to allocate the jobs to the drivers in a reasonable way. For this, there are three considerations, how do we move drivers between landfills? (transitions), how do we satisfy truck and time of day constraints? And how do we manage the supply of cans among the storage yards?

3.3.1 Transitions

In order to determine each transition $\min(RN_i - RT_i)$ for $i = 0$ to $i = n$ is calculated, where RN is the route cost normally, RT is the cost of doing the transition, and n is the number of routes that are unassigned. In the case where pairs are constrained $\min(RN_i - RT_i)$ will always be positive since doing the route normally will always be the same or shorter. However in the case where pairs are non-constrained and can be made between zones the cost of doing a route normally can be less than the cost of doing a transition. Non-transition routes are then simply assigned primarily to fulfill constraints and secondarily in descending order based on distance to the main hub of the company.

3.3.2 Truck and Time Constraints

The two constraints that need to be considered are the time constraint and the truck size constraint. Customers may request a delivery in specifically the AM or PM hours of the day. In order to deal with this constraint routes identified as such are assigned with priority when a driver's schedule is within the time period. In addition, some trucks are restricted as to where they can deliver cans to. This constraint is resolved by identifying routes that require a specific truck and assigning the first driver that truck with a special schedule that services all routes with the constraint and then performs as a regular driver within the zone they end in.

3.3.3 Can Supply Management

A final issue considered is the supply of cans at a given depot. There are two types of can deficits that must be dealt with. First are hard can container deficits which occurs when there are more cans that need to leave the storage location than will be returned to it on a given day. These can be resolved in two stages. The first stage is for a deficit of size up to the number of drivers that will start their day driving to the zone with a deficit. In this stage the calculation for transitioning will be altered to only consider routes with a pickup of the appropriate size can outside the zone with the deficit. In the second stage where the problem cannot be resolved in the initial stage the first driver assigned to the area will be assigned special transitions that both originate and end in the deficit zone, but which leave in order to go to a pickup of the size of the can with a deficit. Next is a soft can deficit which means that the number of returns plus the initial stock of cans at a depot is at least as large as the number of cans of a particular size. This can still be an issue if too many deliveries happen before the pickups for a particular size of can. Even where returns plus the initial stock are at least as large as the number of cans of a particular size a problem may occur if too many deliveries happen before the pickups for a particular size of can. To resolve this the draft schedule is simulated through the day and where conflict arises a swap will be made to fix the problem with priority given to swapping within the same driver's schedule. Due to how our solution works these swaps will not affect the efficiency. When considering whether an issue exists other driver's schedules will be considered to be at a time less than the current simulated time by some amount of minutes L to account for potential differences due to delays.

3.4 Recap

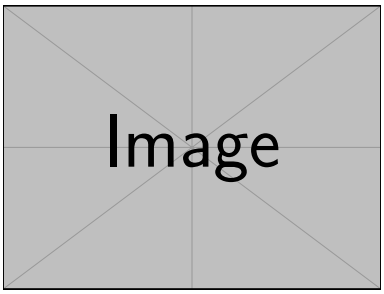
The algorithm for assigning the full schedule are as follows:

1. Receive data for the day and process it into the program.
2. Create pairs with Hungarian algorithm.
3. Assign driver to resolve truck size constraint.

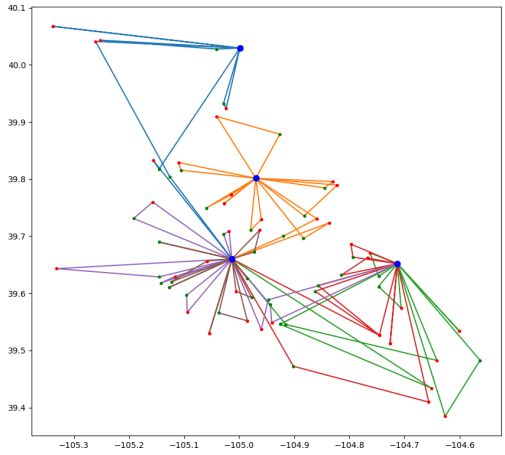
4. Transition to the farthest zone that requires the truck resolving a hard can deficit if applicable.
5. Service all routes within the zone that require the truck and return to 2 if more applicable routes exist in other zones.
6. Finish route as per a normal driver and return to 1 if other truck size constraints exist.
7. Assign next driver to farthest zone with routes.
8. Transition to the zone.
9. If applicable resolve hard can deficits.
10. Assign remaining routes by prioritizing AM/PM constraints and then distance to the main hub.
11. If there are no more routes in the zone return to 6 choosing the next zone with routes remaining.
12. If the driver schedule is full including the transition back to the main hub transition back to the main hub and return to 6.
13. Iterate through the day to resolve any soft can deficits.
14. Output schedules.

4 Results

From an old schedule provided by the client, we ran our algorithm to see how it stacked up to the clients solution **(comparison pending, will be present in final)**. In the following figures, each color is a driver's route.



Route provided by client. Total time spent driving: **PENDING**.



Routes limited to 300 minutes of driving per employee, generated by the algorithm (in 0.41 seconds). Total time spent driving: 1622.8 minutes.

5 Discussion

The program runs very quickly, and it outperforms the client's attempt by **TBD%**. We would have liked to run more data sets to get a more comprehensive report, however, our access to data is limited.

6 Conclusions/Future Work

There are two things that would make this program better, one, is better data. If given actual driving times, the output would do a much better job of reflecting real world conditions. As of now, we actually can't even test how well it's doing.

The other thing that would make this better is a better pair choice reflecting the transitions that will inevitably need to be taken. Right now, pairs are chosen as if drivers can teleport between landfills. While a vast majority of a drivers time is *not* taking transitions, it still seems like we are losing a bit here. In order to truely take them into account however, it seems like a much more painful implementation than what is presented.

In the end, having to take the transitions (as opposed to teleporting between landfills) cost the route about 90 minutes, out of 1622, which is around a 5.5% difference. An improvement could only save us about that much. The obvious next step is to find a bound for the difference between teleportation and our output, however, due to time constraints, we leave it unanswered for now.

7 References

TBD